KDE im Kontext: Open Source Software Entwicklung und öffentliche Güter

Andreas Brand, Ursula Holtgrewe

Beitrag zur Tagung der DGS-Sektionen Arbeits- und Industriesoziologie und Wirtschaftssoziologie uund des Arbeitskreises Politische Ökonomie „Die Wissensökonomie der Wissensgesellschaft“ am 10.-12. Juni 2004 in München

erscheint in: Moldaschl, Manfred/Weber, Hajo (Hg.) 2005: Wissen und Innovation – Beiträge zur Ökonomie der Wissensgesellschaft. Marburg (Metropolis)

1. Vorbemerkung

Open Source oder Freie Software[1] steht für ein Modell technischer Innovation, die nicht oder nicht in erster Linie in Organisationen bzw. auf Märkten stattfindet (z. B. Tuomi 2002; Osterloh u. a. 2004). In Open Source-Projekten entwickeln Hochqualifizierte Software als rechtlich abgesichertes öffentliches Gut auf dezentrale, über das Internet vernetzte, freiwillige und kooperative Weise. Das wohl bekannteste Beispiel ist das Betriebssystem Linux, aber in den letzten Jahren sind neben Entwicklungs- und Programmierwerkzeugen, Internet- und Informationsmanagementprogrammen auch Anwendungen für Nicht-Fachleute (und sogar für Windows) entstanden.

Aus diesem Grund stellt Open Source-Softwareentwicklung wissenschaftlich spannende Fragen nach dem Funktionieren von Innovationsprozessen, den Grenzen des Marktes (Beckert 1997) und seinen Alternativen, und nach den Fragen der gesellschaftlichen Organisation von wissensintensiver und immaterieller Produktion. Wir gehen diesen Fragen am Beispiel des Open-Source-Projekts KDE nach – um so empirisch gegründete, soziologisch gehaltvolle Aussagen darüber zu machen, wie *non-profit*-Innovationen im Ineinandergreifen von technischen Artefakten, sozialen Beziehungen, Normen, Institutionen und Identitäten entstehen und weiter entwickelt werden – und wie die Praxen der Regulierung diese Wissensressourcen wiederum konstituieren.

---

[1] Der Unterschied zwischen Freier Software und Open Source ist politisch-programmatisch. „Freie Software“ im Sinne von „free speech, not free beer“ (Richard Stallman) meint ein normatives politisches Programm freier, selbst bestimmter Zirkulation öffentlicher (Wissens-)Güter. „Open Source“ bezieht sich technisch auf die Verfügbarkeit des Quellcodes, also des Programms in der Art, wie es die Programmiererin geschrieben hat. Um auf einem Computer zu laufen, muss der Quellcode erst in Maschinensprache übersetzt (kompiliert) werden, um an dem Programm zu arbeiten, es zu analysieren und zu verbessern, braucht es den Quellcode. Zur „ideologischen“ Auseinandersetzung s. Holtgrewe 2001.

1.1. OS/FS als Innovationsmodell

In letzter Zeit haben insbesondere Management- und InnovationsforscherInnen die OS/FS-Entwicklung als Innovationsmodell entdeckt, das die Risiken, Ungewissheiten und Dilemmata der Innovation auf Märkten und in formalen Organisationen (Rammert 1988) zu umgehen scheint. Hier zirkulieren Wissen und Produkte frei, hier wird allem Anschein nach von allen Beteiligten entgrenzt, hoch engagiert und unentgeltlich gearbeitet. Die „richtigen" BearbeiterInnen für eine Aufgabe rekrutieren sich überwiegend selbst (Benkler 2002), und Nutzer werden zu Coproduzenten (Raymond 1998; von Krogh u. a. 2003) – in einem höheren und offeneren Maß als man es aus der Literatur über vernetzte Innovationen ansonsten kennt (z. B. von Hippel 1994; Kowol/Krohn 1997; Prahalad/Ramaswamy 2000; Harhoff et al. 2002). Sie liefern Fehlermeldungen und eigene Ergänzungen, Aus- und Umbauten des Produkts und gegenseitige Unterstützung. Somit scheint es, dass FS/OS-Projekte diverse Probleme des Wissensmanagements gelöst haben – und eben dies macht sie für managementorientierte InnovationsforscherInnen so interessant. Jedoch ist ein Spezifikum unübersehbar: Es wird jenseits des Marktes produziert und dafür gelten eigene, institutionalisierte wie informelle Spielregeln.

Hier liegt auch der Charme der FS/OS-Projekte für AutorInnen, die eher gesellschafts- als managementtheoretische Schlüsse ziehen. Sowohl marxistisch (Gorz 2002, Adler 2002, Meretz 2000, Lazzarato 1998) als auch weberianisch (Himanen 2001) und modernisierungstheoretisch (Grassmuck 2000; Tuomi 2002) inspirierte AutorInnen betrachten FS/OS-Projekte daher als eine Konstellation, in der Wissen und IT der Nutzerin als Produzentin neue Macht über die Produktion und Produktionsmittel geben und damit die Institutionen des Marktes und der marktbezogenen Innovation in Frage stellen. Damit teilen diese AutorInnen zwar die Auffassung der Innovations- und ManagementtheoretikerInnen über einen markt- und hierarchiefreien Innovationsmodus, aber sie interpretieren ihn als eine umfassende soziale Alternative zur durchgreifenden Vermarktlichung von Wissen. FS/OS als Modell steht dann exemplarisch für eine Wissensökonomie bzw. -gesellschaft, in der kein Mangel an knappen Ressourcen mehr herrscht, in der die „Hacker-Ethik" selbst bestimmter, intrinsisch motivierter und auf technische Exzellenz gerichteter Arbeit die protestantische Ethik ablöst und der Gebrauchswert gegenüber dem Tauschwert in den Vordergrund rückt.

Von beiden Seiten jedoch, der management- wie der gesellschaftstheoretischen, ist die Diskussion von einem gewisssen Essenzialismus geprägt: Sie konzentriert sich auf die Besonderheiten der Wissensproduktion und die Rolle öffentlicher Güter. Die Güter und

Ressourcen, um die es dabei geht, und entsprechend die Akteure und ihre Motive werden als mit bestimmten Eigenschaften ausgestattet betrachtet, die die FS/OS Produktion ermöglichen. Wir versuchen demgegenüber, eine soziologisch-prozessorientierte Sicht einzunehmen, und den Akzent auf die sozialen Prozesse der Erfindung, Aushandlung und Reflexion zu setzen, in denen die Akteure ihr Feld, die Spielregeln und damit sich selbst erst konstituieren, und damit handelnd spezifizieren, was Regeln und Ressourcen (Giddens 1984) „sind".

1.2 Öffentliche Güter, Wissen und soziale Einbettung

Die Sichtweise öffentlicher Güter antwortet zunächst auf den Befund, dass Märkte Innovationen keineswegs unproblematisch hervorbringen, weil und wenn Innovationen neues Wissen enthalten. Über die Zukunft liefert der Markt keine Informationen, und den Wert des Wissens können Unternehmen weder exakt bestimmen noch vollständig aneignen (Beckert 1997). Das wird vielfach mit der Beschaffenheit von Wissensgütern begründet: Wissen ist nicht teilbar, man kann es gleichzeitig behalten, verkaufen und verschenken. Knapp ist neues Wissen und Wissen über strategisch relevante Perspektiven und Zusammenhänge.

Die Produktion technischer Innovationen als öffentliche Güter ist demnach erklärungsbedürftig. Öffentliche Güter sind Güter, von deren Nutzung niemand (zu vertretbaren Kosten) ausgeschlossen werden kann. In der Systematik von Hess und Ostrom (2003, S. 120) wird zwischen Teilbarkeit und Exklusivität unterschieden (Tab. 1):

		Subtractability	
		low	*high*
Exclusion	*difficult*	**public goods** sunset common knowledge	**common-pool resources** irrigation systems libraries
	easy	**toll or club goods** day-care centers country clubs	**private goods** doughnuts computers

Tab. 1: Einteilung von öffentlichen, privaten, Club- und Allmendegütern (Hess, Ostrom 2003)

Im klassisch-ökonomischen Modell gilt für die *common-pool resources,* also für Ressourcen, die teilbar aber nicht exklusiv sind, die „Tragödie der Allmende" (Hardin 1968): wenn diese für jeden zugänglich sind und die Resultate der Nutzung privat angeeignet werden, werden sie

übernutzt und letztlich zerstört. Trittbrettfahrer, die sich an der Reproduktion nicht beteiligen, lassen sich nicht von der Nutzung ausschließen (Olson 1968). Die „Tragödie der Allmende“ bei der Erstellung öffentlicher Güter führt deswegen entweder zur Einführung zentraler Steuerungs- und Sanktionierungsinstanzen oder zur Etablierung von Eigentumsrechten. Heller (1998) hingegen verweist darauf, dass auch der „Anti-Commons“, d. h. für unsere Zwecke die kommodifizierte Wissensproduktion, ihre Tragödien aufweist: Dann nämlich, wenn die rundum eigentumsrechtlich geschützten Wissensbestände und die damit verbundenen Transaktionskosten zu einer ökonomisch ineffizienten Unternutzung vorhandenen Wissens führen. Gegenüber diesen modellhaften Annahmen hat Ostrom (1990) jedoch deutlich gemacht, dass *communities* aus NutzerInnen sehr wohl in der Lage sind, *common-pool resources* zu bewirtschaften und zu erhalten, indem sie Regeln für Zugang und Nutzung aufstellen und Sanktionen verhängen. Es sind hier also nicht allein *homines oeconomici* am Werke, sondern kompetente Gemeinschaften aus *homines sociologici*, die die Nutzung ihrer Allmenden regulieren.

Um der Frage nachzugehen, *wie* dies bei Wissensgütern geschieht, plädieren Hess und Ostrom (2003) für eine differenziertere Sicht auf öffentliche Güter. Sie unterscheiden auf der Seite expliziten Wissens zwischen den Artefakten (Programme, Texte, E-Mails usw.) und den *facilities* (technischen Werkzeugen, Infrastrukturen, formalen Regulierungen usw.), auf der Seite des impliziten Wissens zwischen Inhalten und Organisationsformen. Das macht es möglich, entlang der Zugänge und Regulierungen zu klären, mit welcher Art von öffentlichen Gütern wir es im vorliegenden Fall zu tun haben. FS/OS-Projekte stellen sich dann – zunächst hypothetisch - als Ensembles aus verschiedenen Arten öffentlicher Güter dar. Zunächst einmal sind die Artefakte, online kostenlos verfügbare Computerprogramme, Betriebssysteme und Werkzeuge nicht teilbar – im Gegenteil, es mag sogar Netzwerkeffekte geben. *Facilities,* wie Internetzugänge, Rechenkapazität und Bandbreite sind teilbar, aber nicht wirklich teuer – und für InhaberInnen von IT-Arbeitsplätzen oder *accounts* an Universitäten fungieren sie eher als Clubgüter. Teilbare *common pool resources* liegen auf der Seite der ProduzentInnen: Zeit, Aufmerksamkeit, Talent als Inhalte (vgl. Bergquist 2003) spielen, wie wir sehen werden, eine Rolle für die Produktion und Nutzung von Innovationen als öffentliche Güter, und werden deswegen durch die Regeln und Praxen der *communities* als Organisationsformen reguliert und strukturiert.

Konstitutiv für die Entwicklung von FS/OS als dezidiert öffentliches Gut ist die Lizenzierung. FS/OS-Lizenzen nutzen die Institutionen geistigen Eigentums, um den offenen Quellcode als

öffentliches Gut nachhaltig abzusichern. Die wichtigste und weitestgehende Lizenz ist die General Public Licence (GPL), die von Richard Stallman als Gründer der Free Software Foundation entwickelt wurde (s. www. gnu.org). Die Pointe der GPL ist ihr „ansteckender" Charakter: Die Lizenz legt fest, dass der Quellcode beliebig weitergegeben und verändert werden darf, aber dass der veränderte Quellcode bei der Weitergabe wieder unter derselben Lizenz veröffentlicht werden muss. Damit wird eine proprietäre Aneignung des offenen Quellcodes mit nachfolgendem Einfordern von Lizenzgebühren ausgeschlossen. Andere Lizenzen dagegen binden Weiterentwickler nicht, oder sie reservieren den ursprünglichen Autoren bestimmte Rechte der Weiterverteilung des modifizierten Code (Working Group 2000). Die Lizenz bestimmt dabei die Art der Softwareerstellung nicht, da auch Firmen ihre Software unter diese Lizenz stellen können und dies auch tun (s.a. Abschnitt 3.).

1.3 „Andere" Ökonomien? Gabentausch und Reputation

Eine eher anthropologische, aber in der Soziologie nicht unbekannte Perspektive nehmen Autoren ein, die in der Erstellung freier oder Open Source-Software eine „andere" Ökonomie des Gabentauschs und der Reputation am Werke sehen. Geradezu klassisch als programmatische Selbstbeschreibung der Szene ist das „Basar"-Modell von Eric Raymond (1998). Auf dem Basar spontaner, selbstorganisierter Kooperation werden Gaben an die Gemeinschaft gegen Reputation getauscht, und die formellen wie informellen Spielregeln sorgen dafür, dass Informationen über Reputation frei zirkulieren können. Damit entspräche FS/OS dem anderen idealtypischen *non-profit*-Innovationsmodell moderner Gesellschaften, nämlich dem der Wissenschaft im Sinne von Merton (1985, vgl. skeptisch Bezroukov 1999).

Auf die Probleme, die Logik des Gabentauschs auf die hochmoderne Wissensproduktion zu übertragen, macht Bergquist (2003) aufmerksam. Sind die materiell-rituellen Gaben etwa bei Mauss (1984) durch Seltenheit, Außeralltäglichkeit und oftmals Zweckfreiheit gekennzeichnet, so stößt die Analogie bei immateriellen öffentlichen Gütern an Grenzen. Im Unterschied zur Wissenschaft mit ihren institutionalisierten Zugangswegen, Informations- und Verteilungskanälen für Reputation ist in unserem Feld noch fraglich, wie Reputation funktioniert, welche Machtbeziehungen, Erwartungen und Verpflichtungen sie stiftet. Der Start eines Projekts, das sich dann als erfolgreich und aufmerksamkeitsträchtig erweist und weitere Beiträge motiviert, kann sicherlich als eine Gabe gelten, für die der Gründer Reputation und damit Macht und Einfluss erwirbt – oder gar, wie bei Linux, ein gewisses Maß an popkultureller Prominenz. Für den Erwerb solcher Reputation ist der Aufwand, ein Projekt anzufangen, notwendig, aber nicht hinreichend. Sie bemisst sich eher am „Erfolg", und für

diesen sind das richtige Timing und die kulturelle Anschlussfähigkeit oftmals entscheidender als die Leistung der ProgrammiererIn. In bestehenden Projekten hingegen basiert der Erwerb von Reputation eher auf dauerhaften, verlässlichen und qualitativ guten Beiträgen.

Ökonomisch betrachtet führt die Frage der Reputation einen weiteren Gütertypus ein: Reputation ist ein knappes Positionsgut, um das konkurriert wird. Die Sichtweise des Gabentauschs aber weist über die Ökonomie hinaus. Sie eröffnet einen Zugang zur Dimension sozialen Sinns und damit zur kognitiven und normativen Konstruktion von Dingen, Regeln und Ressourcen. Das eröffnet die Möglichkeit, einen konstruktivistisch-induktiven Schritt über die Sicht von Hess und Ostrom auf öffentliche Güter hinaus zu machen: Aus der Art der Nutzung erst lässt sich erschließen, um welche Art von Ressourcen es sich jeweils handelt.[2]

Diese erweist sich nur im Prozess des kontextuierten, intersubjektiven Handelns – kurz in der empirischen Beobachtung konkreter Projekte. Gerade in immateriellen, informationstechnisch gestützten und selbstorganisierten Produktionsprozessen kann transparent werden, wie technische Artefakte, soziale Beziehungen, Normen und Werte ineinandergreifen. Um zu untersuchen, wie dies geschieht, wenden wir uns der Organisation und Kooperation des FS/OS-Projekts KDE zu.

2. Das Open Source-Projekt KDE

2.1. Die Organisationsstruktur: Konzentrische Kreise und Module

Das untersuchte Open Source-Projekt KDE (Kool Desktop Environment) entwickelt eine grafische Benutzeroberfläche (Desktop) für Unix-Betriebssysteme. Die Untersuchung des KDE-Projekts erfolgte im DFG-Projekt „Elektronische Arbeitsmärkte“ durch die Auswertung der reichhaltigen Website, die auch biografische Selbstdarstellungen und projektinterne Interviews einiger Mitglieder enthält, die Beobachtung von Mailinglistenkommunikation und Entwicklertagungen, Experteninterviews mit ausgewählten zentralen Entwicklern und durch eine standardisierte quantitative Befragung von 55 KDE-Mitgliedern auf einer Linux-Konferenz.

KDE ist in mancherlei Hinsicht ein recht typisches FS/OS-Projekt. Es wurde Ende 1996 an der Universität Tübingen von drei Studenten gegründet. Das Projekt entwickelte sich von ca. 20 Personen in ihrer Anfangszeit zu einer weltweit verteilten Gruppe von ca. 800-1000 Menschen, die ihren räumlichen Schwerpunkt in Europa, besonders in Deutschland hat. Diese

[2] In der Ökonomie findet sich diese Einsicht bereits bei Penrose : „Strictly speaking, it is never resources themselves that are the ‚inputs‘ in the production process, but only the services that the resources can render. The services yielded by resources are a function of the way in which they are used“ (Penrose 1959/1995: 25).

Gruppe ist demografisch homogen: Der größte Teil der Personen ist männlich und zwischen 20 und 30 Jahre alt und hat einen IT-Hochschulabschluss oder studiert. Die Arbeit im Projekt wird meistens von zu Hause aus in der Freizeit betrieben, der Median der wöchentlichen Arbeitszeit liegt bei zehn Stunden. Die Spannweite der Arbeitszeitangaben liegt jedoch zwischen 0,5 und 90 Stunden pro Woche, was sich durch den hohen Arbeitseinsatz der zentralen Entwickler erklärt. Der Einsatz schwankt auch über die Zeit: Zu bestimmten Zeitpunkten wie der Erstellung einer offiziellen, fehlerfreien und stabil laufenden Version, dem Release, wird besonders viel gearbeitet (Brand, Schmid 2004).

Ein Versionsmanagementsystem im Internet (CVS) bildet das Rückgrat der Softwareentwicklung. Hier werden aktuelle und vergangene Versionen des Quellcodes gespeichert, die man sich zum Arbeiten herunter- und dann wieder hochlädt. Dadurch ist ein räumlich verteiltes und zeitlich versetztes Arbeiten aller Projektmitglieder mit der jeweils aktuellsten Version möglich. Der Lesezugriff auf das Versionsmanagementsystem und die meisten Mailinglisten ist für jeden offen.

Insgesamt haben derzeit 915 Softwareentwickler Schreibzugriff auf das Versionsmanagementsystem, das den bearbeiteten Quellcode, Übersetzungen, Dokumentationen, etc. in 79 Modulen lagert. Eine Auswertung des CVS durch Robles (2004) zeigt, dass seit der Gründung von KDE im Median 11,6 EntwicklerInnen an einem Modul gearbeitet haben. Die Frequenz der Änderungen ist beachtlich: Pro Tag wurden im Median 1155,73 Beiträge ins CVS gestellt, und der Median für Beiträge pro Person pro Jahr liegt bei 458. Die Softwareerstellung findet dabei in kleinen und kleinsten Gruppen von im Durchschnitt vier Personen statt, wobei es auch eine Menge Einpersonen-Projekte gibt. Einzelne Projekte können u.U. 20 oder 30 Personen umfassen (Brand, Schmid 2004).

Der Code und damit das Projekt ist, wie in der Unix-Welt üblich, modular aufgebaut, was wiederum den Koordinationsaufwand reduziert und eine unabhängige, zeitlich versetzte Arbeitsweise ermöglicht. Die Module der Softwareerstellung und damit die (Sub-)Projekte sind konzentrisch strukturiert: Es gibt zentrale Software wie die Kernbibliotheken mit oft benötigtem Quellcode und periphere Anwendungsprogramme, die auf diesen Code zurückgreifen. (Sub-)Projekte haben jeweils einen (Sub-)Projektverantwortlichen (= Maintainer) und weitere Mitglieder. Auch das Gesamtprojekt hat eine konzentrische Struktur: Es gibt einen inneren Kreis von zentralen, stark und andauernd engagierten Projektmitarbeitern und eine Peripherie von Neueinsteigern und gelegentlichen Projektmitgliedern. Neben der Softwareentwicklung gibt es Dokumentations- und

Übersetzungsprojekte, die aber eine nachgeordnete Stellung im Projekt inne haben.[3] Software und Dokumentationen werden in zurzeit 89 Sprachen übersetzt (http://i18n.kde.org/teams/). Der Übergang zwischen den Tätigkeiten Softwareentwicklung und Dokumentation/Übersetzung ist fließend, da Entwickler übersetzen und umgekehrt. Die Projektstruktur bildet sich in den Mailinglisten, dem zentralen Kommunikationsmedium des Projekts, ab.

In KDE sind Ein- und Ausstieg vielfach gleitend. Der Eintritt ist durch Selbstrekrutierung bestimmt. Oft beginnt das Engagement mit der Entdeckung von Fehlern im Programm, die berichtigt werden. Wer mehrmals Fehlerkorrekturen einreicht, bekommt von einem Maintainer Schreibrechte im Versionsmanagementsystem zugeteilt, wenn er oder sie seine Ausdauer und seine Fähigkeiten, d.h. das Schreiben von qualitativ hochwertigem Quellcode, unter Beweis gestellt hat. Es finden sich also einige funktionale Äquivalente der Mitgliedsrolle in Arbeitsorganisationen: Schreibrechte in der Versionsverwaltung und Emailaccounts (@kde.org), die nur an erprobte Mitarbeiter ausgegeben werden. Auch Beiträge von Projektexternen ohne Schreibrechte und Eintrittsambitionen sind willkommen. Sie werden von einem Entwickler überprüft und ggf. eingestellt.

Die Altersstruktur macht deutlich, dass die Mitgliedschaft und der Einsatz im Projekt fluktuieren. Die Projektbeteiligten ziehen sich aus der Softwareentwicklung zurück, wenn Studienende, Erwerbsarbeit, Familiengründung o. ä. die Prioritäten verschieben. Ein Stück weit kann die Arbeit im Projekt als Lebenslaufphänomen gesehen werden.
Über die Projektleitungsfunktionen hinaus haben sich punktuell institutionalisierte Positionen im Projekt entwickelt, wie Infrastrukturposten mit technischer Schreibzugriffsvergabe, Projektleitung in wichtigen und großen Projekten oder um den Releasekoordinator. Diese Positionen werden bei einem Wechsel im Allgemeinen per Kooptation besetzt, d.h. der Vorgänger sorgt für eine Nachfolge. Die NachfolgerInnen werden meistens unter erfahrenen und bewährten Mitgliedern geworben oder muss ein hohes Maß an Eigeninitiative haben, da diese Positionen mit einem hohem Arbeitsaufwand verbunden sind. Solche Entscheidungen erfolgen nach unserer Beobachtung hauptsächlich über informelle Absprachen zwischen Personen aus dem inneren Kreis. Besonders politischen oder mikropolitischen Charakter haben diese Entscheidungen, so weit wir sehen, nicht: Interessenauseinandersetzungen, Fraktionen oder Positionskämpfe sind nicht zu beobachten. Dieses Muster sachbezogener Abstimmung lässt sich auch für andere Entscheidungen, z.B. über zukünftige

[3] In diesem Artikel konzentrieren wir uns auf die Softwareentwickler.

Entwicklungsrichtungen konstatieren. Entsprechend kommen hierarchische Entscheidungen eher punktuell vor, z.B. wenn der Releasekoordinator Termine für offizielle Releases setzt und über noch zulässige Änderungen entscheidet (Brand, Schmid 2004). Die beschriebenen „organisierten" Funktionen wurden dabei vielfach im Modus der „ad-hocracy" (Mintzberg 1996) in Reaktion auf Probleme und Konflikte eingeführt und werden entsprechend modifiziert. Der pragmatische Inkrementalismus der Technikentwicklung wird also im Projekt so weit möglich auch auf die (minimalistische und weitgehend technisch implementierte) Formalstruktur angewandt.

2.2. KDE organisieren: Das Soziale und das Technische

Die Beobachtung wenig konflikthafter oder interessengeleiteter Koordinationsprozesse deckt sich mit den Selbstdeutungen der beteiligten Akteure und verweist auf eine gewichtige normative Grundlage: Matthias Ettrich, KDE-Gründer, schreibt: „In einem technischen Softwareumfeld trifft man selten Entscheidungen; man findet Lösungen [...]. In einer Welt, in der sich im Allgemeinen leicht beweisen lässt, wer recht hat, und alle Parteien offen für logisch korrekte Beweisführungen sind, gibt es naturgemäß wenig Konflikte" (Ettrich 2004, S. 180). Bei näherem Hinsehen wird deutlich, dass das technisch-soziale Gewebe bei der Softwareerstellung in KDE nicht ganz so reibungslos funktioniert. Im folgenden wenden wir uns exemplarischen Konflikten zu, wie sie von Projektbeteiligten in Interviews berichtet und auf Mailinglisten abgebildet wurden, weil hier deutlich wird, wie Regeln und Ressourcen, Leidenschaften und Interessen im Handeln der Beteiligten spezifiziert werden.

(1) Ein Maintainer eines Subprojekts zog sich aus der Softwareentwicklung zurück und ein weiterer Mitarbeiter übernahm seine Nachfolge. Nach ein oder zwei Monaten meldete sich der ehemalige Maintainer wieder und beanspruchte seinen Maintainerposten zurück. Zwischen Nachfolger und Vorgänger entbrannte daraufhin ein Streit, wer der rechtmäßige Maintainer sei. Dieser Streit wurde auf der Mailingliste und im CVS ausgetragen: Die Rivalen löschten abwechselnd den vom je anderen neu eingestellten Quellcode und beschimpften sich auf den Mailinglisten. Dabei spielten auch persönliche Antipathien eine Rolle. Gelöst wurde der Streit durch die Intervention weiterer Subprojektmitarbeiter: Ein dritter Mitarbeiter wurde als neuer Maintainer im Einverständnis der beiden anderen eingesetzt.

Hier werden ungeklärte Ansprüche aufgezeigt, die sich mit persönlichen Konflikten kreuzten. Einerseits gibt es die Norm, dass ein Posteninhaber für einen Nachfolger zu sorgen hat, oder ein dauerhaft unbesetzter Posten dann, wenn sich der Vorgänger auf eine Anfrage nicht meldet, von einem Projektmitarbeiter ungefragt übernommen werden kann. Das sind

meritokratische Spielregeln: Wer Arbeit einbringt, „darf“ auch Entscheidungen treffen, und wer umgekehrt einen Posten innehat, übernimmt die Verpflichtung, die anfallenden Aufgaben zeitnah so zu erledigen, dass andere weiterarbeiten können. Das moralische Recht des Inhabers auf seinen Posten findet eine Grenze am allgemeinen Interesse, dass ein Subprojekt weitergeführt wird. *Wann* aber eine Funktion als „verwaist“, inaktiv oder einstweilen indisponiert gilt, ist die Frage, um die es hier geht. Dass dabei vielbeschäftigte WissensarbeiterInnen zwar über die eigene Kompetenz Bescheid wissen (Benkler 2002), aber nicht immer die eigene Verfügbarkeit und Zeit richtig einschätzen, wird auch aus anderen FS/OS-Projekten berichtet (Moody 2002, S. 172ff.).

(2) Unser zweites Beispiel dreht sich um den Stellenwert öffentlicher Güter und „moralischer“ Autorenrechte. Kurz vor Fertigstellung eines fehlerfreien offiziellen Produkts (Release) legte ein KDE-Mitglied kurz möglicherweise fehlerbehafteten neuen Quellcode im CVS ab. Das kollidierte mit den etablierten Regeln für Releases: Normalerweise besteht zur Herstellung des Release ein Zeitplan, der die noch möglichen Veränderungen im Versionsmanagementsystem abgestuft einschränkt und eine allgemeine Regel darstellt. Ab einem bestimmten Zeitpunkt werden nur noch Fehler beseitigt und keine neuen Funktionen mehr eingebaut. In diesem Fall imitierte der neue Code zudem proprietäre, kommerzielle Programme , was zu rechtlichen Problemen hätte führen können. Der neue, und zu dem fraglichen Zeitpunkt regelwidrige Code wurde dann von den Kernentwicklern herausgenommen, woraufhin der Originalautor allen Quellcode, den er jemals geschrieben hatte, aus dem Versionsmanagementsystem löschte. Die Löschung wurde wiederum rückgängig gemacht und dem Mitglied der Zugang zum System entzogen.

Hier fand jenseits der Lizenz ein Konflikt um „moralische“ Autorenrechte statt: Der freiwilligen, selbst organisierten Leistungserstellung entspricht auch hier der informelle „Kontrakt“, dass, wer die Arbeit macht, auch gewisse Rechte über sein Erzeugnis bzw. dessen Weiterentwicklung hat. Aber: weder technisch noch moralisch bzw. rechtlich gehört aus der Sicht der Kernentwickler das Recht dazu, einmal aufgenommene Beiträge zurückzuziehen. Diese sind sozusagen irreversibel der Gemeinschaft übereignet. Die Reaktion des gekränkten Autors in diesem Fall reklamierte sozusagen individuelle Autorenrechte an einem Punkt, an dem diese nicht mehr durchzusetzen waren.

Eine Folge dieses Vorgangs war dann später die Einführung von informellen Positionen neben dem formellen Releasemanager, die kurz vor dem Release alle Quellcodeveränderungen in wichtigen Subprojekten autorisieren. Mit der Anpassung der Projektstruktur und der

Etablierung von Entscheidungspositionen auf Zeit werden solche Konflikte verhindert, aber auch der Releaseplan wird besser überwacht und neue Fehler durch Korrekturen vermieden.
(3) Der dritte exemplarische Fall dreht sich um den Import quasi philosophischer Debatten und interkultureller Auseinandersetzungen. Ein lang engagierter, US-amerikanischer Mitarbeiter vertrat auf Mailinglisten ausführlich eine libertär-ideologische Weltsicht und kritisierte immer wieder den Ablauf von Entscheidungen, z.B. über die Übernahme von Aufgaben oder die Größe der Entscheidungskompetenz des Releasekoordinators. Andere Mitarbeiter waren an solchen Debatten eigentlich nicht interessiert, sahen sich aber genötigt, die Diskussionen zu führen. Sie sahen wegen der öffentlichen Zugänglichkeit und Transparenz der Mailinglisten die Reputation des Projekts gefährdet, wenn sie diese Aussagen nicht im Sinne der von der Mehrheit geteilten Normen relativierten. Der Mitarbeiter hat sich nach einigen solcher Debatten aus dem Projekt zurückgezogen und programmiert außerhalb des Projekts an KDE-kompatibler Software weiter.
Hier werden nicht nur interkulturelle Debatten deutlich, sondern die Durchsetzung von Relevanzregeln auf Mailinglisten. Philosophische oder politische Auseinandersetzungen auf themenzentrierten Mailinglisten werden für sachfremd („*off-topic*“) erklärt. Diese Art der Kommunikation ist nach unseren Beobachtungen nur für eine Minderheit ein Anreiz (die übrigens eigens dafür eine wenig populäre *off-topic*-Mailingliste eingerichtet bekam). Für die Mehrheit gehört die technisch-sachlich-freundliche, auf den Code fokussierte Kommunikation zu den Voraussetzungen, sich auf die intrinsisch motivierte Tätigkeit des Programmierens beschränken zu können – und dieses technisch unterstützte Aufmerksamkeitsmanagement kann sich wiederum auf subjektive Dispositionen und professionelle Normen stützen, die in einer technischen Ausbildung einsozialisiert werden. Auf der anderen Seite spielt die externe Reputation eine Rolle: Kontroverse Aussagen auf Mailinglisten (oder auch in Kommentarzeilen im Quellcode selbst) können sich auf den Ruf oder das Image eines Projekts auswirken und so Sponsoren oder potentielle Entwickler abschrecken.
(4) Auch unser viertes Beispiel einer Kontroverse zwischen einem Entwickler und einem Endnutzer verweist auf die Kommunikation mit der Umwelt des Projekts: Der Entwickler fühlte sich durch Kritik oder Vorschläge des Endnutzers „genervt“ und reagierte entsprechend in einer Email. Die Beschwerde des Endnutzers auf einem KDE-internen Newsboard über unfreundliche Verkehrsformen führte zu einer Entschuldigung des Entwicklers. Dies illustriert die Beziehung zwischen den beiden Gruppen: Generell stellt sich KDE gegenüber Nutzern ausgesprochen offen und freundlich dar, weil diese immer potentielle Mitarbeiter sind. Gerade

eine verbreitete grafische Benutzeroberfläche aber hat eine breite Nutzerbasis aus technischen Laien ohne Programmierkenntnisse. Auch diese Endnutzer sind auf der einen Seite als Softwarefehlerberichter und Ideengeber für Verbesserungen gern gesehen, so dass man versucht, durch eine vielfältige Außendarstellung des Entwicklungsprozesses mit Blogs der Entwickler (Tagebüchern mit Forencharakter) oder über Mailinglisten einen Wissensaustausch herbeizuführen. Auf der anderen Seite herrscht im Projekt die Norm der Selbstorganisation bzw. selbstbestimmten Übernahme von Aufgaben: Bloße Benutzerwünsche werden wesentlich weniger gern gesehen als Lösungen.

2.3 Aufmerksamkeit als *common pool resource*

Deutlich wird an den vier Fällen zunächst das Ineinandergreifen von technischen Werkzeugen und Kommunikation. Weil es um den Code als „Medium und Maschine“ (Esposito 1993) geht, werden nicht nur die Mailinglisten, sondern auch das Versionsmanagementsystem zur Arena und zum Medium sozialer Auseinandersetzungen. Das Hochladen und Löschen von Code ist keineswegs eine rein technische Operation. Es kann das Äquivalent für lautstarke Auseinandersetzungen und türenknallende Abgänge sein.

Bei allem Konsens über den Vorrang technischen Funktionierens und sachlicher Problemlösung spielen unterhalb der rechtlich abgesicherten Ebene öffentlicher Güter die „moralischen“ Autorenrechte, Ansprüche und Erwartungen eine Rolle, die die Beteiligten mit ihren Leistungen und Gaben verbinden. Eine grundlegende Norm besteht offensichtlich darin, dass, wer eine Leistung erbringt, auch gewisse Entscheidungsrechte über deren Weiterentwicklung behält. Mit diesen Rechten verbunden aber ist für die Maintainer die Verpflichtung, sich auch um ihr Projekt zu kümmern und Zeit und Aufmerksamkeit für Beiträge Anderer aufzubringen – oder den Posten zu übergeben. Sozial schwierig wird dies, wenn Verantwortliche ihre zeitlichen und aufmerksamkeitsbezogenen Kapazitäten überschätzen oder wenn rivalisierende Ansprüche aufkommen. Nicht zu den moralischen Autorenrechten gehört das Recht, Beiträge zurückzuziehen. Wäre dies technisch oder sozial möglich, so wäre das Projekt insgesamt gefährdet. Vorrang vor den moralischen Autorenrechten hat also die Sicherstellung, dass ggf. von anderen weiter gearbeitet werden kann.

Die allseitige Öffentlichkeit und Transparenz von Code und Kommunikation dient zwar der Zugänglichkeit für neue MitarbeiterInnen und Beiträge. Andererseits generiert die umfassende Information und lebhafte Kommunikation leicht Informationsüberflutungen. Die KDE-Mitarbeiter müssen das bekannte Dilemma der Wissensarbeit austarieren: Einerseits aus

umfassenden Informationen die relevanten herausfiltern, andererseits genug Kontextinformationen aufnehmen, um auf dem Laufenden zu sein, Einfluss nehmen zu können und Entscheidungen zu antizipieren (Franck 1998, S. 68).

Zur Bewältigung dieses Balanceaktes dienen die technischen und normativen Regulierungen. Zeit und Aufmerksamkeit im Projekt erweisen sich in der Tat als die *common pool resources*, auf die es ankommt. Damit alle möglichst viel davon dem Programmieren auf dem aktuellen Stand zuwenden können, besteht die Norm der zeitnahen Bearbeitung, an denen die moralischen Autorenrechte und die Entscheidungsrechte einzelner ihre Grenze finden. Und die Verpflichtungen der Maintainerrolle richten sich darauf, dies zu gewährleisten.

Die Normen der freundlichen und sachbezogenen Kommunikation bewirtschaften die Aufmerksamkeit und das Engagement aktueller und potenzieller MitarbeiterInnen. Für die Mehrheit sind philosophische und politische Debatten auf den Mailinglisten „off-topic", und zudem können kontroverse Statements oder Unfreundlichkeiten potenzielle EntwicklerInnen und weitere Öffentlichkeiten abschrecken.[4]

Sanktionen treten in Kraft, wenn diese Normen verletzt werden. Je nach Situation werden Ignorieren, Ironie, „Flaming" bzw. Beleidigungen als textbasierte Mailinglistensanktionen eingesetzt, wobei kollektives Ignorieren einem Ausschluss gleichkommen kann (vgl. Döhring 2001). Das Maximum an Sanktion, das sparsam eingesetzt wird, ist dann der technische Ausschluss aus Mailingliste oder Versionsmanagementsystem.

2.4 Reputation im Projekt

Diese Betrachtung macht es uns möglich, die Funktionsweise von Reputation im Projekt genauer zu bestimmen. Projektbeteiligte formulieren als Kriterien für Reputation die kompetente Handhabung des Ineinandergreifens von Technik und Sozialität im Interesse des Projekts (Brand, Schmid 2004): Bei der Produktion zählen qualitativ hochwertige Software und die freiwillige Übernahme dringlicher Aufgaben. Kommunikative Präsenz in Gestalt der Beteiligung in Mailinglisten oder im Chat ist wichtig, und dabei zählen Erfahrung und Freundlichkeit. Außerdem arbeiten Personen mit Reputation besonders an aufwändigen und schwierigen Aufgaben oder an zentraler Software. Nicht überraschend entsteht ein Mertonscher Matthäus-Effekt (Merton 1968): Ein Projektmitarbeiter mit hoher Reputation bekommt oft Posten angeboten, wodurch seine Reputation weiter erhöht wird (Brand, Schmid 2004) – mitsamt dem Risiko der Überlastung (Bezroukov 1999). Reputation funktioniert rekursiv: Sie dient gleichzeitig als Anreiz, als Bestätigung bestehender Normen im Projekt

[4] Diese Normen schließen an die Regeln der „Netiquette", des guten Benehmens in der virtuellen Kommunikation mit Fremden an, die in den meisten Internet-Kommunikationszusammenhängen gelten.

und als Werkzeug zur Aufmerksamkeitsfilterung. Auf Mailinglisten kann man wichtige Informationen nach der Reputation der Absender selektieren. In der Überprüfung technischer Beiträge im Versionsmanagementsystem funktioniert es umgekehrt: Je mehr Reputation eine Person hat, desto weniger werden ihre Beiträge vom Maintainer oder anderen kontrolliert. Vertrauen spart hier Aufwand.

Obwohl unsere Beobachtungen vielfach mit denen von Eric Raymond (1998) übereinstimmen, ziehen wir den umgekehrten Schluss: Raymond meint, die Transparenz der Informationsflüsse in OS-Projekten diene der effizienten Kommunikation und Verteilung von Reputation. Es scheint jedoch, dass die Reputation auf mehreren Ebenen eher der Aufmerksamkeitsregulierung angesichts tendenzieller Informationsüberflutung dient. Sie dient als Filter und als Bewertungskriterium. In Konflikten bezieht sich die Regulierung dabei vorrangig aufdie nachhaltige Gewährleistung befriedigender Arbeits- und Entfaltungsmöglichkeiten für die Gesamtheit aktueller und potenzieller MitarbeiterInnen. Hier stehen Ansprüche aufgrund vergangener Leistungen zurück.

Common pool resources sind also die Arbeitszeit und Aufmerksamkeit der aktuellen EntwicklerInnen im Projekt. Sie sind teilbar und beziehen sich auf lebendige Arbeit –werden aber mit der Offenheit der Projekte für neue Mitarbeiter um die Allmendegüter potenzieller Mit- und Zuarbeit erweitert. Auch diese kann man im Sinne der *tragedy of the commons* übernutzen. Hier regulieren Normen, Habitus und Reputation – gestützt durch technische Werkzeuge.

3. KDE im Kontext: Zwischen sozialer Bewegung und Geschäftsmodellen

Technisch steht KDE in engem Zusammenhang mit anderen Projekten und Programmen. Es bildet als grafische Benutzeroberfläche ganz materiell die Schnittstelle zwischen einem Betriebssystem, meist Linux, und den Anwendungsprogrammen, die teils zum Projekt gehören (der Browser Konqueror, das Büropaket Koffice usw.), teils unabhängig davon installiert werden können. Grafische Benutzeroberflächen sind auch für die Distributionen wichtig, die aus Linux und Anwendungsprogrammen, Entwicklungsumgebungen sowie Installationswerkzeugen quasi „Pakete“ schnüren und verkaufen bzw. weitere Dienstleistungen anbieten. Sie machen z.B. Linux erst für ein weiteres Publikum aus technischen Laien benutzbar. Auch als Programmierwerkzeuge, Compiler usw. setzen EntwicklerInnen Open-Source-Programme ein. Der offene Quellcode macht es möglich, Code auch zwischen Projekten auszutauschen und einzubauen. KDE benutzt z.B. das Fehlerberichtssystem (Bugzilla) des Open Source-Browsers Mozilla, das an die

Gegebenheiten von KDE angepasst wurde und darauf ausgerichtet ist, Fehlermeldungen von großen Nutzerzahlen ein Stück weit automatisiert weiterzugeben. Die Beziehung zwischen KDE und anderen Projekten ist demnach freundlich-kooperativ.

Darüber hinaus kann aber auch von Open Source im Sinne einer sozialen Bewegung die Rede sein, die eine normative Grundlage geteilter Überzeugungen und Solidarität hat (vgl. Della Porta, Diani 1999: 14ff, zu FS/OS speziell Zimmermann 2004). Sie basiert auf der Ansicht, dass Wissen frei zugänglich sein sollte. Einerseits sind dies professionelle Normen technischer und wissenschaftlicher Fachkulturen, andererseits kommt bei den Programmatikern freier Software um die Free Software Foundation (FSF) eine Emphase hinzu, die auf dem Gebrauchswert und der intelligenten Nutzung von Wissensgütern zielt. Diese sind Gegenstände des sozialen Austauschs, der kreativen Aneignung und Weiterentwicklung.

Dies ist auch die normative Grundlage der GPL, an der die Ambitionen sozialer Bewegung im Feld am deutlichsten werden. Ihr „ansteckender“ Charakter sucht Wissensgüter und Kreativität gewissermaßen nachhaltig zu bewirtschaften, indem sie die weiterhin produktive und kreative Nachwelt an die Bedingungen offener Nutzung bindet – eine Selbstbindung an Freiheit, und ein Versuch, sich und die Nachwelt auf Vielfalt festzulegen.

Jedoch funktioniert die GPL in der sozialen Arena der FS/OS nicht allein als normativer und programmatischer Bezugspunkt. Sie sichert – neuerdings in Deutschland auch gerichtsgetestet (vgl www.heise.de/newsticker/meldung/49377) – den Code als öffentliches Gut ab (vgl. O‘Mahony 2003a) und kann auch zur Grundlage wirtschaftlichen Handelns werden: Wenn ein Unternehmen Programme nicht aneignen kann, so kann sein Konkurrent das auch nicht (vgl. Holtgrewe/Werle 2001). Wirtschaftliche Interessen im Feld sind also durchaus legitim. Open-Source-Geschäftsmodelle basieren etwa auf kundenspezifischen Anpassungen der Programme, auf Dienstleistungen, auf dem Verkauf komplementärer Produkte oder, wie bei großen Hardwareherstellern, auf der Erschließung von strategischen Alternativen zum Microsoft-Monopol oder der Etablierung offener Standards (vgl. Working Group 2000; Pal/Madanmohan 2002, Brügge et al. 2004)

Wie prozessiert nun KDE die Normen, Interessen und Ambitionen der unterschiedlichen Akteure im Feld? Die Rollen der Lizenz, der Rechte und des Code zwischen normativer Einbindung und Geschäftsmodellen werden an den Anfängen von KDE deutlich (vgl. Moody 2002, S. 252ff.). KDE stand von Anfang an unter der GPL, basierte aber auf kommerziell entwickelten, frei verfügbaren Programmbibliotheken ohne GPL (Qt der norwegischen Firma

Trolltech). Die Free Software Foundation-Vertreter hielten die Verbindung von proprietärem Code und GPL-Code für rechtlich bedenklich. Einige Vertreter gründeten daher Gnome, ein vollständig GPL-lizenziertes konkurrierendes Desktop-Projekt. Es gab ausgiebige Debatten zwischen KDE, FSF und Trolltech, die zur Neuentwicklung einer Open-Source-kompatiblen Lizenz und schließlich zur Doppellizenzierung der Qt-Bibliotheken unter der GPL-Lizenz und einer kommerziellen Lizenz führten. Des weiteren wurde KDE gegen eine Übernahme oder Insolvenz der Firma abgesichert.

Dass es immer wieder gelingt, Firmen von freieren Lizenzkonstruktionen zu überzeugen und Code zu „befreien“, macht deutlich, dass die institutionelle Absicherung der Produktion jenseits des Marktes nicht nur EntwicklerInnen motiviert (die auf ihr „Eigentum“ verzichten, aber das mittels der GPL auf Gegenseitigkeit tun), sondern auch für Firmen funktioniert und ein ebenes Spielfeld für Dienstleistungen und proprietäre Anpassungen schafft.

Abgesichert wird auch die rechtliche Konstruktion durch die Bedeutung von Reputation im Feld: Lizenzüberschreitungen werden, wenn sie nicht stillschweigend ausgeräumt werden, alsbald durch negative Diskussionen über die betreffende Firma in wichtigen Internetforen bzw. Kommunikationsplattformen oder durch viele direkte Emails an die Firma sanktioniert (vgl. O’Mahony 2003a). Projekte und Unternehmen haben also reziproke Beziehungen, und die Unternehmen akzeptieren die Regeln der Open Source-Bewegung. Als Gegenleistung subventionieren Firmen auch die *non-profit*-Aktivitäten: Sie spenden für die Organisation von Konferenzen und lagern Homepage, Mailinglisten und Dateiablagesystem auf ihren Servern. Außerdem sind einige (im Sommer 2004 ca. 3, früher bis zu 10) KDE-Entwickler bei Firmen angestellt, die auch während ihrer Arbeitszeit an KDE mitarbeiten, und durch ihr hohes Zeitbudget und ihren Einsatz vielfach auch wichtige Funktionen übernehmen.

Diese gegenseitigen Abhängigkeiten werden ständig beobachtet und gegebenenfalls angepasst: Von den Tausch- und Kooperationsbeziehungen im Feld geht ein gewissermaßen isomorpher (DiMaggio/Powell 1991) Druck aus, formale Organisationen zu bilden: Bei KDE ist KDE e.V. nicht nur für die Verwaltung der Sponsorengelder zuständig, sondern er hält alle wichtigen Rechte (copyright-, Markenrechte) zur Sicherung des gemeinsamen Produkts gegen proprietäre Aneignung (vgl. O’Mahony 2003a, b; Brand, Schmid 2004). Versuche von projektnahen Firmen, wiederum den Verein zu dominieren, kamen bereits vor. Deswegen wurden neuerdings Schutzfunktionen gegen eine „Übernahme“ in das Vereinsstatut eingefügt.

Für Konflikte mit Firmen über Lizenzen und Mißbräuche zeichnet sich eine Aufgabenteilung zwischen Free Software Foundation und KDE ab. Zunächst wird mit der entsprechenden

Firma verhandelt (dies übernehmen Mitglieder des innneren Kreises des Projekts), die informelle Sanktionierung übernehmen die Foren der FS/OS-Szene, die rechtliche Verfolgung soll ggf. der FSF übertragen werden (Brand, Schmid 2004).

Als soziale Bewegung also hat die FS/OS-Arena bestimmte Besonderheiten: Zwar gibt es „ideologisch" unterschiedliche Positionen und durchaus lautstarke Auseinandersetzungen, aber diese schlagen nur begrenzt auf die Entwicklung in den Projekten und die Kooperation der Akteure im Feld durch. Weit reichende Bekenntnisse und ideologische *commitments* werden auch von den Aktiven nicht verlangt. Der gemeinsame institutionelle und normative Nenner scheint stabil genug, um hier einen relativ breiten Konsens zu ermöglichen. Dieser jedoch besteht zentral im Focus auf dem Produkt und der technisch-fachlichen Exzellenz. Das technische Funktionieren also gewinnt einen normativen, ingenieurkulturell und habituell verankerten Eigenwert, der wiederum die weiterreichenden Ansprüche auf Freiheit und Kreativität auf den pragmatischen Boden bringt.

4. Ist es Arbeit?

Die FS/OS Projekte also erfinden Organisationen und Institutionen neu auf vielfältige, einerseits lösungsorientierte, andererseits ad-hocratisch improvisierte Weise. Als analog gemischt erweisen sich die Motive und Dispositionen der EntwicklerInnen selbst. Die Motivation der EntwicklerInnen ist empirisch gut erforscht. In den letzten Jahren sind einige Befragungen von OS/FS-EntwicklerInnen durchgeführt worden, die ein ziemlich konsistentes Bild ergeben (Ghosh u. a. 2002; Lakhani/Wolf 2003; Hertel u. a. 2003).[5] An Motiven für das Engagement in FS/OS-Projekten rangieren in den Befragungen intellektuelle Anregung, das Lernen, die Entwicklung und das Teilen der eigenen Fähigkeiten übereinstimmend an erster Stelle. Hoffnungen auf verbesserte Arbeitsmarktchancen spielen eine mittlere Rolle. Normative und politische Überzeugungen, dass Code frei sein sollte (Lakhani/Wolf 2003) oder Software nicht proprietär (Ghosh u. a. 2002), motivieren etwa ein Drittel der EntwicklerInnen (einen ausführlicheren Überblick geben Luthiger 2004 und Holtgrewe 2004). Aufschlussreich sind die Befunde von Hertel u. a. (2003): Sie stellen fest, dass der zeitliche Einsatz – wie das Engagement in anderen sozialen Bewegungen – signifikant am stärksten mit der Identifikation der Befragten als Aktive, d. h. als Linux-EntwicklerInnen und

[5] Lakhani und Wolf(2003) haben 684 EntwicklerInnen befragt, die aus einer Zufallsauswahl von Projekten auf der Open Soure-Plattform Sourceforge identifiziert worden. Robles u. a. (2001, n = 5478) und Ghosh u. a. (2002, n = 2784), Hertel u. a. (2003, n= 141, davon 69 aktive Entwickler) haben ihren Fragebogen ins Internet gestellt und die Einladung zur Teilnahme an eine Auswahl relevanter Mailinglisten und News-Foren geschickt.

EntwicklerInnen spezifischer Subsysteme zusammenhängt. Einflussreich ist weiterhin die Erwartung persönlicher Vorteile und ein Faktor, den Hertel u.a. als „lack of concern for time losses“ beschreiben. Wenn das Programmieren aber schlicht Spaß macht, die aufgewendete Zeit gerade nicht gerechnet wird, dann erschließen sich die Akteure hier ein Handlungsfeld, in dem sie es sich gewissermaßen leisten, auf das Kalkül von Aufwand und Ergebnis, von Kosten und Nutzen zu verzichten. Der spezifische „Anreiz“ besteht dann darin, sich vom kreativen Prozess absorbieren zu lassen, also den *homo oeconomicus* hinter sich zu lassen.
Damit stellt sich die Frage, was FS/OS-Entwicklung mit Erwerbsarbeit, ehrenamtlicher Arbeit und Eigenarbeit gemeinsam hat. Zunächst lässt sich KDE als Non-Profit-Gebilde fassen: Es ist ein freiwilliges, selbstverwaltetes, nicht staatliches Projekt, das keinen Gewinn erzielt und aufs Gemeinwohl orientiert ist (vgl. Bentem 2001, S. 29f.).[6] Jedoch ist der Verein KDE e. V., der die klassische Non-Profit-Organisation bildet, nur ein Teil des Projekts. Er wurde gegründet, um die Außenbeziehungen zu Unternehmen zu managen, und die Vorsitzenden spielen eine relativ wichtige Rolle z.B. bei der Organisation von KDE-Konferenzen. Die eigentliche Softwareentwicklung ist jedoch davon abgekoppelt. Hier werden Entscheidungen informell, und ad-hoc getroffen und Positionen informell besetzt.
Für die ProjektmitarbeiterInnen ist die Beteiligung überwiegend ein Hobby, das individuell und meist von zu Hause aus betrieben wird und Programme auch für den Eigenbedarf entwickelt. Damit steht diese Art der Tätigkeit zwischen organisiertem Ehrenamt und individuellem Hobby bzw. Eigenarbeit (vgl. Mutz 2002), und man kann sagen, dass sie für die Beteiligten die Vorteile von beiden Betätigungen verbindet: Das technische Hobby ist in einen größeren, kooperativen Zusammenhang mit offenen Grenzen zur Professionalisierung eingebunden, das Ehrenamt zeitlich, sachlich und sozial flexibel. Technisch und sozial institutionalisiert und an ingenieurprofessionelle Fachkulturen anschließend, gelingt es den FS/OS-Projekten offensichtlich, freiwillige Arbeit so zu organisieren, dass die Schwierigkeiten des klassischen Ehrenamtes mit individualisierten Lebens- und Arbeitsformen umgangen werden (vgl. Schumacher 2003, S. 71f.). Hier kann man sein Engagement variieren, Koordinations- und Gewährleistungsarbeiten ein gutes Stück automatisieren, Reputation für technisch-fachliche Exzellenz erwerben, normative und sozial bewegte *commitments* verfolgen und soziale Kontakte zwischen virtueller Kommunikation und *face-to-face* dosieren: Ehrenämter für hoch individualisierte *homines fabri* gewissermaßen.

---

[6] Andere Projekte werden in stärkerem Maße von Gruppen betrieben, in denen FirmenvertreterInnen als solche agieren. Sie fallen eher unter die Organisationsform überbetrieblicher F&E-Kooperationen.

Die globale Offenheit dieser Kooperationsform funktioniert jedoch offensichtlich nur für bestimmte Gruppen in bestimmten Kontexten: es sind ganz überwiegend jüngere männliche Hochqualifizierte, die sich hier engagieren. Insbesondere der Frauenanteil liegt nach allen vorliegenden Untersuchungen mit 1 – 3 % deutlich unter dem etwa der Informatikstudierenden. Im Lichte der Diskussion um Technik und Geschlecht kann man annehmen, dass die spezifische Passion für mutmaßlich „sachliche" Kriterien technischer Exzellenz (vgl. Hacker 1989), die Verkehrsformen der Reputationsökonomie und das bei aller Offenheit und Hilfsbereitschaft doch ein Stück weit elitäre Selbstverständnis an männliche Vergemeinschaftungsformen anknüpfen, die man vom Hobbyistenzirkel bis zur Wissenschaftler-*community* kennt, und diese pfadabhängig reproduzieren.

Wie aber sehen die Subjektdispositionen aus, die wir hier im Feld antreffen, wenn wir sie mit der Diskussion um subjektivierte und informatisierte Arbeit vergleichen (s. die Beiträge in Schönberger/Springer 2003)? Zwar sind individuelle Gründe und Motive, sich zu engagieren höchst vielfältig, aber sie unterscheiden sich deutlich von jenen Subjekten, die wie die berühmten ArbeitskraftunternehmerInnen marktförmige Subjektivierung an sich selbst vollziehen (Voß/Pongratz 1998) oder den betrieblichen Anforderungen nur hinterher rennen (Glißmann 2002). Die Tätigkeit ist sehr weitgehend selbstbestimmt und wird auch selbst rationalisiert sowie kontrolliert.[7]

Die Selbstrationalisierung aber hat einen spezifischen, technisch-gebrauchswertorientierten und dezidiert nicht-unternehmerischen Akzent: Sie bezieht sich, wenn das Programmieren hinreichenden Spaß macht, nicht auf den zeitlichen oder anstrengungsmäßigen Einsatz (Hertel u. a. 2003). Um aber diese Intrinsik realisieren zu können, sucht man per Aufmerksamkeitsmanagement und mit technischen Werkzeugen Doppelarbeit zu vermeiden und Arbeit punktgenau einzusetzen.

Für zentral halten wir das Motiv der „Selbstoptimierung", wie es Pongratz (2001) skizziert hat: Die selbstbestimmte Suche nach Herausforderungen, Bewährungs- und Lernchancen, das sich aber hier gerade nicht in marktbezogene Orientierungen fügt. Hier finden sich wieder Ähnlichkeiten zum Lernen und der Tätigkeit in Vereinen oder in informellen Gruppen (vgl. Oshege 2001) und dann, wenn man die Räume und institutionellen Voraussetzungen selbstbestimmter Tätigkeit zu erweitern sucht, zum Lernen in sozialen Bewegungen.

---

[7] Eine Selbstökonomisierung ist nicht flächendeckend zu beobachten, da in den einschlägigen Befragungen (s.o.) das Motiv der Qualifizierung oder des „signalling" für den Arbeitsmarkt zwar vorhanden, aber nicht dominant ist (s.a. Brand, Schmid 2004).

Die ingenieurprofessionellen Habitus, die in die normativen Grundlagen der FS/OS-Entwicklung eingehen, entstammen zwar der Erwerbsarbeitssphäre (und dem Bildungssystem), aber sie liegen offensichtlich quer zur Vermarktlichung. Was auch in der Erwerbsarbeit zu Widersprüchen führt (Glißmann 2002), kann hier ein gutes Stück jenseits des Marktes selbstbestimmt und von vornherein gebrauchswertorientiert entfaltet werden. Es ist also auch für diesen Fall sinnvoll „Arbeit“ in einem Kontinuum zwischen öffentlicher und privater Tätigkeit und formellen und informellen Kontexten einzuordnen (vgl. Taylor 2004): Je nach Projekten und Kontexten bilden FS/OS-Projekte hybride Zusammensetzungen aus professionellen Kooperationen, Freiwilligenorganisationen und „Alternativprojekten“ sozialer Bewegungen.

5. Fazit: Hybride öffentlicher Güter

An der Empirie wird deutlich, dass das Ensemble aus öffentlichen Gütern, rechtlicher Absicherung und flexibel-freiwilliger Kooperation nicht von selbst funktioniert. Freie Software bildet ein Ensemble aus allen vier Arten von Gütern bei Hess und Ostrom (2003) – und noch einigen mehr. Wir finden mit dem Code rechtlich abgesicherte öffentliche Güter (obwohl deren rechtliche Sicherheit zurzeit umkämpft ist, wie die Klage der Firma SCO gegen IBM deutlich macht); es gibt private Güter wie Computer, und es gibt die zentralen *common pool resources* der Zeit und Aufmerksamkeit im Projekt. Genau gesagt, sind Zeit und Aufmerksamkeit zunächst teilbare, exklusive private Güter über die jedeR selbst verfügt. Als *common pool resource* bewirtschaftet wird die Bereitschaft aktueller und potenzieller EntwicklerInnen, diese auf Beiträge zum Projekt zu verwenden, und diese Bereitschaft hängt an der Gewährleistung der Möglichkeit, dort Dinge zu tun, die Spaß machen. Als selbstregulierte Clubgüter fungieren dann die Nutzung, die Lern- und Gemeinschaftschancen der *non-profit*-Softwareentwicklung, denn diese setzen die entsprechenden Fähigkeiten voraus.

Das Zusammenspiel dieser Güter, ihrer Erstellung und ihrer Reproduktion wird wie gesehen in einem engen Wechselspiel technischer und sozialer Regulierung gewährleistet, und in den alltäglichen Praxen und Auseinandersetzungen werden, ganz wie Hess und Ostrom das sehen, an zentraler Stelle die *common pool resources* reguliert.

Erfolgreiche Open Source-Projekte bewegen sich damit in und zwischen den genuin wissensgesellschaftlichen Spannungsfeldern von Knappheit und Reichtum, Komplexitätsaufbau und -reduktion und managen diese Spannungsfelder:

- Die institutionell innovative rechtliche Absicherung öffentlicher Güter ermöglicht die Vernetzung und (normative) Integration heterogener kollektiver Akteure von der sozialen Bewegung bis zum Unternehmen;
- die technische Unterstützung vernetzter und asynchroner Zusammenarbeit und die Transparenz der Produkte und Beiträge ermöglicht die flexible Beteiligung;
- die technischen und sozialen „Werkzeuge“ ermöglichen es, sich dem Spaß und der Intrinsik des Programmierens hinzugeben, d. h. die Toleranz für zeitlichen Aufwand und Umwege sprechen nicht gegen einen rationalen Umgang mit den eigenen Arbeitseinsätzen;
- Die thematische Fokussierung, die Selbstkontrolle und die im Feld geltenden Normen regulieren dabei die Bereitschaft, Aufmerksamkeit und Zeit einzusetzen. Sie sind professionell-habituell bei den Teilnehmern verankert, was wiederum (erleichtert durch deren relative soziale Homogenität) die Offenheit und Selbstorganisation ermöglicht.

Mit dem Akzent auf der Kontextgebundenheit und dem prozessualen, immer wieder herzustellenden Charakter dieser Ensembles aus Code, Werkzeugen, Praxen, Leidenschaften und Interessen wird deutlich, dass es sich hier bislang schwerlich um umfassend-deterministische Produktivkraftentfaltung handelt. Es ist jedoch aussichtsreich, auch in anderen Kontexten technischer, ökonomischer und sozialer Innovation von den Projekten zu lernen, dass sich soziale Alternativen zu Märkten und Hierarchien finden lassen, dass diese aber auch ihre spezifischen Voraussetzungen haben und soziale Schließungen produzieren.

6. Literatur

Adler, P. (2002): Critical in the Name of Whom and What? Organization 9 (3), 387 - 395.

Beckert, J. (1997): Grenzen des Marktes. Frankfurt/Main, New York: Campus.

Benkler, Y. (2002): Coase's Penguin, or Linux and the Nature of the Firm. Yale Law Journal 112.

Bentem, N. (2001): Freiwillige Vereinigungen – Eine vergessene Größe im Dritten Sektor. In: B. Strob (Hrsg.): Vereintes Lernen. Regionale Lernkulturen und Vereinslandschaften in den alten und neuen Bundesländern. Münster: Waxmann.

Bergquist, M. (2003): Open-Source Software Development as Gift Culture: Work and Identity Formation in an Internet Community. In: C. Garsten; H. Wulff (Hrsg.): New Technologies at Work. People, Screens, and Social Virtuality. Oxford, New York: Berg, 223-241.

Bezroukov, N. (1999a): Open Source Software Development as a Special Type of Academic Research. Critique of Vulgar Raymondism. First Monday 4. http://www.firstmonday.org/issues/issue4_10/bezroukov/index.html, 10.8.2000.

Brand, A.; Schmid, A. (2004): A case study of an Open Source project. http://www.soz.uni-frankfurt.de/arbeitslehre/pelm/docs/FALLSTUDIE%20Open%20Source%20V1.pdf, 28.7.2004.

Brügge, B.; et al. (2004): Open-Source-Software. Eine ökonomische und technische Analyse. Berlin u. a.: Springer.

Della Porta, D.; Diani, M. (1999): Social movements: an introduction. Oxford: Blackwell Publishers.

DiMaggio, P. J.; Powell, W. W. (1991): The iron cage revisited: Institutional isomorphism and collective rationality. In: W. W. Powell; P. J. DiMaggio (Hrsg.): The New Institutionalism in Organizational Analysis. Chicago, London: Chicago UP, 63 - 82.

Döhring, N. (2001): Belohnungen und Bestrafungen im Netz: Verhaltenskontrolle in Chat-Foren. Gruppendynamik und Organisationsberatung – Zeitschrift für angewandte Sozialpsychologie, 32 (2), 109-143

Esposito, E. (1993). Der Computer als Medium und Maschine. Zeitschrift für Soziologie, 22, 338 – 354.

Ettrich, M. (2004): Koordination und Kommunikation in Open-Source-Projekten. In: R. Gehring; B. Lutterbeck (Hrsg.): Open Source Jahrbuch 2004. Zwischen Softwareentwicklung und Gesellschaftsmodell. Berlin: Lehmanns Media, 179-192.

Franck, G. (1998): Ökonomie der Aufmerksamkeit. Ein Entwurf. München, Wien: Hanser.

Ghosh, R.; Glott, R.; Krieger, B.; Robles, G. (2002): Free/Libre and Open Source Software: Survey and Study. Deliverable D 18: Final Report, Part 4: Survey of Developers, Maastricht (international Institute of Infonomics), http://floss1.infonomics.nl/finalreport/FLOSSfinal-4.pdf, 25.7.2002.

Giddens, A. (1984): Die Konstitution der Gesellschaft. Grundzüge einer Theorie der Strukturierung. Frankfurt: Campus, 1988.

Glißmann, W. (2002): Der neue Zugriff auf das ganze Individuum. Wie kann ich mein Interesse behaupten? In: M. Moldaschl; G. G. Voss (Hrsg.): Subjektivierung von Arbeit. München, Mering: Hampp, 241-259.

Gorz, A. (2002): Welches Wissen? Welche Gesellschaft? In: Heinrich-Böll-Stiftung (Hrsg.): Gut zu wissen. Links zur Wissensgesellschaft. Münster: Westfälisches Dampfboot, 14-35.

Grassmuck, V. (2000): Open Source - Betriebssystem für eine freiheitliche Gesellschaft. Vortrag auf der Tagung "Freie Software - Ein Modell für die Bürgergesellschaft", Evangelische Akademie Tutzing 31. Mai - 1. Juni 2000. http ://www.waste.informatik.hu-berlin.de/grassmuck/texts/oss-tutzing-5-00.html, 7.9.2004.

Hacker, S. (1989): Pleasure, power and technology. Some tales of gender, engineering, and the co-operative workplace. Boston u. a.: Unwin Hyman.

Hardin, G. (1968): The Tragedy of the Commons. Science 162, 1243-1248.

Harhoff, D.; Henkel, J.; Hippel, E. von (2002): Profiting from voluntary information spillovers: How users benefit from freely revealing their innovations. MIT Sloan School of Management Working Paper (July), Revised May, 2002. Cambridge, Mass.

Heller, M. A. (1998): The Tragedy of the Anticommons: Property in the Transition from Marx to Markets. Harvard Law Review 111 (3), 622-688.

Hertel, G.; Niedner, S.; Herrmann, S. (2003): Motivation of software developers in open source projects: An internet-based survey of contributors to the Linux kernel. Research Policy 32 (7), 1159-1177.

Hess, C.; Ostrom, E. (2003): Ideas, Artifacts, and Facilities: Information as a Common-Pool Resource. Law and Contemporary Problems 111 (1&2), 111-146.

Himanen, P. (2001): The hacker ethic and the spirit of the information age. New York: Random House.

Hippel, E. v. (1994): "Sticky Information" and the locus of problem solving: Implications for innovation. Management Science 40: 429-439.

Holtgrewe, U. (2001): Kreativität als Norm - zum Erfolg verdammt? Open-Source-Software zwischen sozialer Bewegung und technischer Innovation. In: J. Allmendinger (Hrsg.): Gute Gesellschaft? Verhandlungen des 30. Kongresses der Deutschen Gesellschaft für Soziologie in Köln 2000. Opladen: Leske, 399-424.

Holtgrewe, U. (2004): Heterogene Ingenieure - Open Source und Freie Software zwischen technischer und sozialer Innovation. In: R. A. Gehring; B. Lutterbeck (Hrsg.): Open Source Jahrbuch 2004. Zwischen Softwareentwicklung und Gesellschaftsmodell. Berlin: Lehmanns Media, 339-351.

Holtgrewe, U.; Werle, R. (2001): De-commodifying software? Open Source software between business strategy and social movement. Science Studies 14, 43-65.Kowol, U.; Krohn, W. (1997): Modernisierungsdynamik und Innovationslethargie. In: B. Blättel-Mink; O. Renn (Hrsg.): Zwischen Akteur und System. Die Organisierung von Innovation. Opladen: Westdeutscher Verl., 39-65.

Krogh, G. von ; Spaeth, S.; Lakhani, K. (2003): Community, joining and specialization in open source software innovation: A case study. Research Policy 32 (7), 1217-1241.

Lakhani, K.; Wolf, B. (2003): Why Hackers Do What They Do: Understanding Motivation and Effort in Free/Open Source Software Projects.
http://opensource.mit.edu/papers/lakhaniwolf.pdf, 23.11.2003

Lazzarato, M. (1998): Immaterielle Arbeit. Gesellschaftliche Tätigkeit unter den Bedingungen des Postfordismus. In: T. Negri; M. Lazzarato; P. Virno (Hrsg.): Umherschweifende Produzenten. Immaterielle Arbeit und Subversion. Berlin: id-Verlag, 39-52.

Luthiger, B. (2004): Alles aus Spaß? Zur Motivation von Open-Source-Entwicklern. In: R. A. Gehring; B. Lutterbeck (Hrsg.): Open Source Jahrbuch 2004. Zwischen Softwareentwicklung und Gesellschaftsmodell. Berlin: Lehmanns Media, 93-106.

Mauss, M. (1984): Die Gabe: Form und Funktion des Austauschs in archaischen Gesellschaften. Frankfurt/Main: Suhrkamp.

Meretz, S. (2000): LINUX & CO. Freie Software - Ideen für eine andere Gesellschaft. Version 1.01, letzte Änderung: 03.07.2000. http://www.kritische-informatik.de/fsrevol.htm, 7.9.2004.

Merton, R. K. (1968): The Matthew Effect in Science. Science 159 (3810), 56-63.

Merton, R. K. (1985): Entwicklung und Wandel von Forschungsinteressen. Aufsätze zur Wissenschaftssoziologie. Frankfurt/Main: Suhrkamp.

Mintzberg, H. (1996): The Adhocracy. In: H. Mintzberg; J. Quinn (Hrsg.): The Strategy Process. Englewood Cliffs, N. J: Prentice Hall, 679-693.

Moody, G. (2002): Rebel Code. Linux and the open source revolution, London: Penguin.

Mutz, G. (2002): Pluralisierung und Entgrenzung in der Erwerbsarbeit, im Bürgerengagement und in der Eigenarbeit. Arbeit 11 (4), 21-32.

O'Mahony, S. (2003a): Guarding the commons: How community managed software projects protect their work. Research Policy 32 (7), (July 2003): 1179-1198.

O'Mahony, S. (2003b): Non-Profit Foundations and their Role in Community-Firm Software Collaboration. In: Making Sense of the Bazaar: Perspectives on Open Source and Free Software. Boston: O'Reilly & Associates Publications.

Olson, M. (1968): Die Logik des kollektiven Handelns: Kollektivgüter und die Theorie der Gruppen. Tübingen: Mohr.

Oshege, V. (2001): Lernpotentiale in freiwilligen Vereinigungen. Ausgewählte Ergebnisse einer qualitativen Studie. In: B. Strob (Hrsg.): Vereintes Lernen. Regionale Lernkulturen und Vereinslandschaften in den alten und neuen Bundesländern. Münster: Waxmann

Osterloh, M.; Rota, S.; Kuster, B. (2004): Open-Source-Softwareproduktion: Ein neues Innovationsmodell? In: R. A. Gehring; B. Lutterbeck (Hrsg.): Open Source Jahrbuch 2004. Zwischen Softwareentwicklung und Gesellschaftsmodell. Berlin: Lehmanns Media, 121-137.

Ostrom, E. (1990): Governing the Commons. New York: Cambridge UP.

Pal, N.; Madanmohan, T.R. (2002): Competing on Open Source: Strategies and Practise. Working Paper. http://opensource.mit.edu/online-papers.php, 15.7.2004.

Pongratz, H. J. (2001): Das Beste herausholen. Die Mitbestimmung 6, 26-29.

Prahalad, C. K.; Ramaswamy, V. (2000): Wenn Kundenkompetenz das Geschäftsmodell mitbestimmt. Harvard Business Manager 4, 64-76.

Rammert, W. (1988): Das Innovationsdilemma. Technikentwicklung im Unternehmen. Opladen: Westdeutscher Verlag.

Raymond, E. (1998): The cathedral and the bazaar. First Monday 3 (3). http:// www.firstmonday.org/issues/issue3_3raymond/index.html, 7.9.2004.

Robles, Gregorio (2004): (BETA3) CVS Analysis for the KDE project. http://libresoft.dat.escet.urjc.es/cvsanal/kde3-cvs/index.php?menu=Statistics, 20.8.2004.

Schönberger, K., Springer, S. (Hrsg.): Subjektivierte Arbeit. Mensch, Organisation und Technik in einer entgrenzten Arbeitswelt. Frankfurt: Campus.

Schumacher, U. (2003): Lohn und Sinn. Individuelle Kombinationen von Erwerbsarbeit und freiwilligem Engagement. Opladen: Leske und Budrich

Taylor, R. (2004): Extending conceptual boundaries: work, voluntary work and employment. Work, employment and society. Vol. 18 (1): 29-49.

Tuomi, I. (2002): Networks of innovation. Change and meaning in the age of the Internet. Oxford: Oxford UP.

Voss, G. G.; Pongratz, H. J. (1998): Der Arbeitskraftunternehmer. Eine neue Grundform der Ware Arbeitskraft? Kölner Zeitschrift für Soziologie und Sozialpsychologie 50. 131-158.

Working Group on Libre Software (2000): Open Source: Information society opportunities for Europe? Version 1.2. http://eu.conecta.it/paper.pdf, 7.9.2004

Zimmermann, T. (2004): Open Source und Freie Software – soziale Bewegung im virtuellen Raum? In: R. A. Gehring, B. Lutterbeck (Hrsg.): Open Source Jahrbuch 2004. Zwischen Softwareentwicklung und Gesellschaftsmodell. Berlin: Lehmanns Media.